

PATENT APPLICATION

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Re the Application of

Masakazu ISOMURA

Application No.: 10/801,642

Filed: March 17, 2004

Docket No.: 119118

For: METHOD FOR REFERRING TO ADDRESS OF VECTOR DATA AND VECTOR
PROCESSOR

CLAIM FOR PRIORITY

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

The benefit of the filing date of the following prior foreign application filed in the following foreign country is hereby requested for the above-identified patent application and the priority provided in 35 U.S.C. §119 is hereby claimed:

Japanese Patent Application No. 2003-203856 filed on July 30, 2003

In support of this claim, a certified copy of said original foreign application:

☒ is filed herewith.

It is requested that the file of this application be marked to indicate that the requirements of 35 U.S.C. §119 have been fulfilled and that the Patent and Trademark Office kindly acknowledge receipt of this document.

Respectfully submitted,

James A. Oliff
Registration No. 27,075

Thomas J. Pardini
Registration No. 30,411

JAO:TJP/mlo

Date: April 12, 2004

OLIFF & BERRIDGE, PLC
P.O. Box 19928
Alexandria, Virginia 22320
Telephone: (703) 836-6400

DEPOSIT ACCOUNT USE
AUTHORIZATION
Please grant any extension
necessary for entry;
Charge any fee due to our
Deposit Account No. 15-0461

日 本 国 特 許 庁
JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office.

出 願 年 月 日 2 0 0 3 年 7 月 3 0 日
Date of Application:

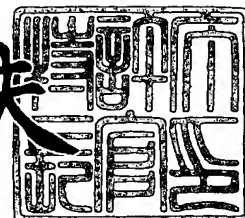
出 願 番 号 特 願 2 0 0 3 - 2 0 3 8 5 6
Application Number:
[ST. 10/C] : [J P 2 0 0 3 - 2 0 3 8 5 6]

出 願 人 セイコーエプソン株式会社
Applicant(s):

2 0 0 4 年 4 月 2 日

特許庁長官
Commissioner,
Japan Patent Office

今 井 康 夫



出証番号 出証特 2 0 0 4 - 3 0 2 7 4 9 9

【書類名】 特許願

【整理番号】 J0100879

【あて先】 特許庁長官殿

【国際特許分類】 G06F 15/00

【発明者】

【住所又は居所】 長野県諏訪市大和 3 丁目 3 番 5 号 セイコーエプソン株式会社内

【氏名】 磯村 政一

【特許出願人】

【識別番号】 000002369

【氏名又は名称】 セイコーエプソン株式会社

【代理人】

【識別番号】 100095728

【弁理士】

【氏名又は名称】 上柳 雅誉

【連絡先】 0 2 6 6 - 5 2 - 3 5 2 8

【選任した代理人】

【識別番号】 100107076

【弁理士】

【氏名又は名称】 藤網 英吉

【選任した代理人】

【識別番号】 100107261

【弁理士】

【氏名又は名称】 須澤 修

【先の出願に基づく優先権主張】

【出願番号】 特願2003- 92381

【出願日】 平成15年 3月28日

【手数料の表示】

【予納台帳番号】 013044

【納付金額】 21,000円

【提出物件の目録】

【物件名】 明細書 1

【物件名】 図面 1

【物件名】 要約書 1

【包括委任状番号】 0109826

【プルーフの要否】 要

【書類名】 明細書

【発明の名称】 ベクトルデータのアドレス参照方法およびベクトルプロセッサ

【特許請求の範囲】

【請求項 1】 指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うアドレス参照方法であって、

指標ベクトルの要素を格納する要素格納用レジスタを複数領域に分割し、各領域に所定のコードを格納し、前記指標ベクトルの要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴とするベクトルデータのアドレス参照方法。

【請求項 2】 前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、

前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、参照先のメモリアドレスであるターゲットアドレスを算出することを特徴とする請求項 1 記載のベクトルデータのアドレス参照方法。

【請求項 3】 前記指標ベクトルをベクトルレジスタに格納し、該ベクトルレジスタの各要素レジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴とする請求項 2 記載のベクトルデータのアドレス参照方法。

【請求項 4】 前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、

前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、参照先のメモリアドレスであるターゲットアドレスを算出し、算出されたターゲットアドレスを新たな参照基準アドレスとすることを特徴とする請求項 1 記載のベクトルデータのアドレス参照方法。

【請求項 5】 前記相対アドレスを示すコードを前記要素格納用レジスタとしてのスカラーレジスタに格納し、該スカラーレジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴とする請求項 4 記載のベクトル

ルデータのアドレス参照方法。

【請求項 6】 前記ベクトルデータの読み出しあるいは書き込みに関するベクトル命令の実行中に、前記分割された各領域のうち、選択する領域を動的に変化させることを特徴とする請求項 1～5 のいずれかに記載のベクトルデータのアドレス参照方法。

【請求項 7】 前記要素格納用レジスタの前記分割された各領域のうち、選択する領域を指定するための指定パターンを所定のレジスタに格納し、該指定パターンに基づいて前記分割された領域を指定することにより、所定の指標ベクトルを生成することを特徴とする請求項 6 記載のベクトルデータのアドレス参照方法。

【請求項 8】 第 1 の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第 1 のレジスタと、第 2 の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第 2 のレジスタとを用意し、第 1 および第 2 のレジスタそれぞれにおける所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴とする請求項 1～7 のいずれかに記載のベクトルデータのアドレス参照方法。

【請求項 9】 指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うベクトルプロセッサであって、

指標ベクトルの要素を格納する要素格納用レジスタを備え、

該要素格納用レジスタは、複数領域に分割されて各領域に所定のコードが格納され、

該要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能であることを特徴とするベクトルプロセッサ。

【請求項 10】 第 1 の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第 1 のレジスタと、第 2 の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第 2 のレジスタとを備え、第 1 および第 2 のレジスタそれぞれにおける所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能であることを特徴とする請求項 9 記載のベクトルプロセッサ。

【発明の詳細な説明】

【0001】**【発明の属する技術分野】**

本発明は、ベクトルデータの読み出しあるいは書き込みにおけるアドレス参照方法およびベクトルプロセッサに関する。

【0002】**【従来の技術】**

従来、画像処理等において、不規則にメモリ上のデータの読み出しあるいは書き込むことが行われており、そのような処理を効率的に行うべく、種々の方法が提案されている。

例えば、画像処理において、1枚の画像データの中で、特定のブロックのデータをメモリからレジスタに格納する場合、ブロックの一行分の画素データをレジスタに格納した後、ブロックの次行の画素データを読み込むために、メモリ上の所定長離れたアドレスが参照される。

【0003】

このように、複雑なアドレス参照を行うための方法として、参考文献「スーパーコンピュータ」（オーム社、長島重夫、田中義一著）に記載された技術が知られている。

本文献においては、多次元配列データをメモリからベクトルレジスタに読み込んだり、ベクトルレジスタからメモリに書き込んだりする場合に、間接指標ベクトル参照を用いる技術が開示されている。

【0004】

間接指標ベクトル参照とは、メモリ上の参照するアドレス順序を格納したリスト（指標ベクトル）を用意しておき、そのリストを順に参照することによって、間接的にメモリ上の所定のアドレスを参照する方式である。

このような間接指標ベクトル参照を用いることにより、メモリ上の複雑なアドレス参照を行うことが可能となる。

【0005】**【非特許文献1】**

長島重夫、田中義一著「スーパーコンピュータ」オーム社、p. 35

- 41

【0006】

【発明が解決しようとする課題】

しかしながら、従来の技術においては、指標ベクトルをベクトルレジスタに格納しておく必要があるため、以下のような問題が生じていた。

第1に、指標ベクトルを用意しておく手順が必要となるため、プログラムのコードサイズや処理サイクル数が増加してしまう。

【0007】

第2に、用意しておく指標ベクトルが増加すると、指標ベクトルを格納しておくベクトルレジスタも増加することから、本来の演算処理に用いるレジスタリソースが不足し、演算効率の低下を招いてしまう。

本発明の課題は、間接指標ベクトル参照をより効率的に行うことである。

【0008】

【課題を解決するための手段】

以上の課題を解決するため、本発明は、

指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うアドレス参照方法であって、指標ベクトルの要素を、格納する要素格納用レジスタ（例えば、図4のレジスタファイル40におけるインデックスベクトルを格納しているレジスタ）を複数領域（例えば、上位および下位の領域）に分割し、各領域に所定のコードを格納し、前記指標ベクトルの要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴としている。

【0009】

例えば、発明の実施の形態における命令コード内の“エクステンション”によって、インデックスレジスタにおいて分割されたいずれかの領域を選択する。そして、選択された領域のコードによって、指標ベクトル（インデックスベクトル）の要素が決定され、所定のメモリアドレスを指定する指標ベクトルが生成される。

【0010】

また、前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレス（例えば、ベースアドレス）に対する相対アドレスを示すコードを格納し、前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコード（例えば、上位16ビットあるいは下位16ビット等）と、該基準アドレスとに基づいて、参照先のメモリアドレスであるターゲットアドレス（メモリ上の参照アドレス）を算出することを特徴としている。

【0011】

この方法は、発明の実施の形態におけるインデックス修飾アドレッシングに関連する。

また、要素格納用レジスタにおける分割された領域のいずれを選択するかは、命令コード中で指定したり、選択する領域を指定する指定パターンをレジスタ等に記憶しておき、記憶された指定パターンを入力して領域を指定するといったことが可能である。

【0012】

また、前記指標ベクトルをベクトルレジスタに格納し、該ベクトルレジスタの各要素レジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴としている。

また、前記要素格納用レジスタの各領域は、前記指標ベクトルの参照基準アドレスに対する相対アドレスを示すコードを格納し、前記要素格納用レジスタそれぞれの分割された領域のうち、選択された領域のコードと、該基準アドレスとに基づいて、参照先のメモリアドレスであるターゲットアドレスを算出し、算出されたターゲットアドレスを新たな参照基準アドレスとすることを特徴としている。

【0013】

この方法は、発明の実施の形態におけるポスト・レジスタ・アップデート・アドレッシングに関連する。

また、前記相対アドレスを示すコードを前記要素格納用レジスタとしてのスカラーレジスタに格納し、該スカラーレジスタを分割した各領域に、前記相対アドレスを示すコードを格納することを特徴としている。

【0014】

また、前記ベクトルデータの読み出しあるいは書き込みに関するベクトル命令の実行中に、前記分割された各領域のうち、選択する領域を動的に変化させることを特徴としている。

また、前記要素格納用レジスタの前記分割された各領域のうち、選択する領域を指定するための指定パターンを所定のレジスタに格納し、該指定パターンに基づいて前記分割された領域を指定することにより、所定の指標ベクトルを生成することを特徴としている。

【0015】

また、第1の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第1のレジスタ(例えば、発明の実施の形態中の「インデックスレジスタ」と、第2の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第2のレジスタ(例えば、図9のサブインデックスレジスタ141, 142)とを用意し、第1および第2のレジスタそれぞれにおける所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能とすることを特徴としている。

【0016】

また、本発明は、

指標ベクトルを用いて、ベクトルデータの読み出しあるいは書き込みにおけるメモリアドレスの参照を行うベクトルプロセッサであって、指標ベクトルの要素を格納する要素格納用レジスタ(例えば、発明の実施の形態中の「インデックスレジスタ」の要素レジスタ)を備え、該要素格納用レジスタは、複数領域に分割されて各領域に所定のコードが格納され、該要素格納用レジスタそれぞれの所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能であることを特徴としている。

【0017】

また、第1の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第1のレジスタ(例えば、発明の実施の形態中の「インデックスレジスタ」と、第2の指標ベクトルの要素が格納された前記要素格納用レジスタを含む第2のレジスタ(例えば、図9のサブインデックスレジスタ141, 142)とを備え

、第1および第2のレジスタそれぞれにおける所定領域に格納されたコードを用いて、複数の指標ベクトルを生成可能であることを特徴としている。

【0018】

ここで、第1のレジスタおよび第2のレジスタの要素格納用レジスタにおける分割された領域のいずれを選択するかは、命令コード中で指定したり、選択する領域を指定する指定パターンをレジスタ等に記憶しておき、記憶された指定パターンを入力して領域を指定するといったことが可能である。

本発明によれば、ベクトルレジスタの要素レジスタあるいはスカラーレジスタを複数の領域に分割して、それぞれの領域に指標ベクトルの要素となる所定のコードを格納する。そして、分割した領域のいずれかを選択し、その領域に格納されたコードを用いて、所定の指標ベクトルを取得することとしている。

【0019】

したがって、1つの指標ベクトルを格納するレジスタの領域に、実質的に複数のインデックスベクトルを格納できることとなり、レジスタリソースを効率的に使用することが可能となる。

【0020】

【発明の実施の形態】

以下、図を参照して本発明に係るベクトルプロセッサの実施の形態を説明する。

（第1の実施の形態）

本発明に係るベクトルプロセッサは、指標ベクトルを格納するベクトルレジスタあるいはスカラーレジスタにおいて、ベクトルレジスタの要素レジスタあるいはスカラーレジスタを分割して使用し、アドレッシング機能を拡張している。

【0021】

したがって、初めに、このような機能を実現するための基本となる考え方について説明する。なお、本発明は、ベクトルプロセッサにおけるロード命令およびストア命令に関するものであるため、これらを中心に説明する。

ベクトルプロセッサにおいて、ロード命令あるいはストア命令のコードタイプには、アドレッシングモード（アドレス参照の方法）に応じて、以下の3種類が

規定されている。

【0022】

図1は、ロード命令あるいはストア命令のフォーマットを示す図であり、(a)は、ベース相対アドレッシングに対応するLS0タイプ、(b)は、ポスト・オフセット・アップデート・アドレッシングに対応するLS1タイプ、(c)は、インデックス修飾アドレッシングおよびポスト・レジスタ・アップデート・アドレッシングに対応するLS2タイプを示している。

【0023】

図1において、LS0タイプはスカラーデータのロード命令およびストア命令に対応しており、LS1タイプおよびLS2タイプは、スカラーデータおよびベクトルデータのロード命令およびストア命令に対応している。

本発明は、ベクトルデータのロード命令およびストア命令を取り扱うものであり、インデックス修飾アドレッシングおよびポスト・レジスタ・アップデート・アドレッシングに関するものであるため、LS2タイプについて説明する。

【0024】

図1(c)において、LS2タイプのフォーマットには、オペコード (opcode)、デスティネーション (dst)、ベース (base)、リピートアmount (rptamt)、エクステンション (extension)、インデックス (index) の6つのフィールドが含まれている。

オペコードは、命令の内容を示すフィールドであり、ロード命令あるいはストア命令のいずれかを示すコードが含まれている。

【0025】

デスティネーションは、ロードあるいはストアの対象となるデータが格納されたレジスタのアドレスを指定するフィールドである。即ち、ロード命令の場合、メモリから読み出したデータを書き込むレジスタのアドレスであり、ストア命令の場合、メモリに書き込むデータを読み出すレジスタのアドレスである。

ベースは、基準となるメモリアドレスが格納されたレジスタのアドレスを指定するフィールドである。

【0026】

リピートアmountは、ベクトル命令における命令の繰り返し回数（要素データ数）を示すフィールドであり、命令繰り返し回数が“1”である場合はスカラー命令、“1”以外である場合はベクトル命令となる。

エクステンションは、機能拡張用に用意された5ビットのフィールドであり、本発明においては、後述のように、エクステンションを利用して、分割した要素レジスタのアドレスを指定する。

【0027】

インデックスは、ベースに示されたアドレスに対し、どのような修飾を施すか、即ち、ベースに示されたアドレスから参照するアドレス順序を示すフィールドである。本発明においては、後述のように、インデックスを利用して、アドレス修飾の拡張仕様を規定する。

なお、インデックスは、上述の指標ベクトルが格納されたベクトルレジスタあるいはスカラーレジスタ（以下、指標ベクトルが格納されたレジスタを「インデックスレジスタ」と言う。）を指定しており、インデックスを参照することによって、間接指標ベクトル参照が行われる。

【0028】

続いて、本発明におけるアドレス修飾の拡張仕様について具体的に説明する。

図2は、インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。なお、図2においては、アドレス修飾の拡張仕様を説明するため、C言語に準じた記述スタイルのプログラミング例を併せて示しており、後述の図3においても同様である。

【0029】

図2において、エクステンションが“000”である場合、インデックスとして指定されたレジスタ（インデックスレジスタ）の内容（指標ベクトル）を符号付きで解釈し、アドレス修飾を行う。

具体的には、ベースに示されたレジスタの値を常に基準とし、その値にインデックスに示されるアドレスを順次加算することにより、アドレス修飾を行う。

【0030】

また、エクステンションが“001”である場合、インデックスとして指定されたレジスタの下位16ビットを符号付きで解釈し、アドレス修飾を行う。

また、エクステンションが“010”である場合、インデックスとして指定されたレジスタの上位16ビットを符号付きで解釈し、アドレス修飾を行う。

さらに、エクステンションが“011”である場合、インデックスとして指定されたレジスタの下位16ビットおよび上位16ビットをそれぞれ符号付きで解釈し、交互にアドレス修飾を行う。具体的には、ベクトル命令における繰り返しの実行順序が偶数である場合、下位16ビットによってアドレス修飾を行い、奇数である場合、上位16ビットによってアドレス修飾を行う。

【0031】

次に、ポスト・レジスタ・アップデート・アドレッシングの拡張仕様について説明する。

図3は、ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【0032】

図3において、エクステンションが“000”である場合、インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレスの更新を行う。

具体的には、ベースに指定されたレジスタの値を所定のレジスタ（図3中のパラメータTの値を格納するレジスタ）に一旦格納し、そのレジスタの値を更新しながらアドレス修飾を行う。

【0033】

また、エクステンションが“001”である場合、インデックスとして指定されたレジスタの下位16ビットを符号付きで解釈し、アドレスの更新を行う。

また、エクステンションが“010”である場合、インデックスとして指定されたレジスタの上位16ビットを符号付きで解釈し、アドレスの更新を行う。

さらに、エクステンションが“011”である場合、インデックスとして指定されたレジスタの下位16ビットおよび上位16ビットをそれぞれ符号付きで解釈し、交互にアドレスの更新を行う。

【0034】

ここで、エクステンションが“000”～“011”までの場合、ベクトル命令における命令の繰り返し回数の間、インデックスとして指定された内容に応じてアドレス修飾が行われ、繰り返し回数が終了した後、ベースに指定されたレジスタの値が更新される。

したがって、複数のロード命令あるいはストア命令を用いて、繰り返しロードあるいはストアを行う場合、ベースに指定されたレジスタが自動的に更新されるので、命令発行の都度、ベースのアドレスを別の命令で更新する必要がなく、直ちに命令の実行を行うことが可能である。

【0035】

また、図3において、エクステンションが“100”～“111”の場合、エクステンションが“000”～“011”の場合とそれぞれ同様にアドレスの更新を行う。ただし、エクステンションが“100”～“111”の場合、命令の繰り返し回数が終了した後、ベースに指定されたレジスタの値を更新することなく命令の実行を終了する。

【0036】

次に、本実施の形態に係るベクトルプロセッサの構成を説明する。

図4は、本実施の形態に係るベクトルプロセッサ1の構成を示す図である。

図4において、ベクトルプロセッサ1は、メモリ10と、メモリ制御部20と、命令フェッチ部30と、レジスタファイル40と、ロードユニット50と、ストアユニット60と、演算ユニット70とを含んで構成される。

【0037】

メモリ10は、ベクトルプロセッサ1に与えられる命令コードおよび演算対象となるデータを記憶している。

メモリ制御部20は、メモリ10に対するアクセス、即ち、データの読み出しや書き込みを制御する。例えば、メモリ制御部20は、ロードユニット50あるいはストアユニット60によって指定されたメモリ10のアドレスからデータを読み出したり、メモリ10から読み出されたデータをレジスタファイル40に出力したりする。

【0038】

命令フェッチ部30は、メモリ制御部20を介して、メモリ10から命令コードをフェッチし、一時的に記憶する。

レジスタファイル40は、32個のスカラレジスタSR0～SR31と、8個の要素レジスタからなる8個のベクトルレジスタVR0～VR7とを含んで構成され、メモリ10から読み出されたデータおよび演算結果を一時的に記憶する。なお、以下の説明において、ベクトルレジスタの要素レジスタおよびスカラレジスタは、32ビットの幅を持つものとして説明する。

【0039】

ロードユニット50は、命令フェッチ部30に記憶された命令コードがロード命令である場合に、メモリ10から命令コードあるいはデータを読み出す処理を行う。

ストアユニット60は、命令フェッチ部30に記憶された命令コードがストア命令である場合に、メモリ10にデータを書き込む処理を行う。

【0040】

演算ユニット70は、命令フェッチ部30に記憶された命令コードが所定の演算命令である場合に、レジスタファイル40に記憶された所定データを対象として演算処理を行う。

ここで、ロードユニット50の構成について、詳細に説明する。

図5は、ロードユニット50の内部構成を示すブロック図である。

【0041】

図5において、ロードユニット50は、命令パイプライン制御部51と、インデックスレジスタ決定回路52と、デステイネーションレジスタ決定回路53と、アドレス演算回路54と、パイプラインレジスタ（PR）55、56と、レジスタ57とを含んで構成される。

命令パイプライン制御部51は、ロードユニット50全体を制御するものである。

【0042】

インデックスレジスタ決定回路52は、命令コードのインデックスフィールド

に基づいて、指標ベクトルが格納されたインデックスレジスタを選択する信号（インデックスレジスタ選択信号）を生成する。

なお、インデックス修飾アドレッシングの場合、基準として指定されたアドレス（ベースアドレス）を保持し、そのアドレスに対する各要素の相対アドレスを指定することで参照アドレスを生成する。即ち、インデックス修飾アドレッシングの場合、要素ごとに相対アドレスを指定するため、インデックスレジスタとしてベクトルレジスタが指定される。

【0043】

一方、ポスト・レジスタ・アップデート・アドレッシングの場合、基準として指定されたアドレス（ベースアドレス）を参照アドレスとすると共に、そのアドレスに対する相対アドレスを指定してベースアドレスが更新される。そして、その更新されたベースアドレスに対する、次の要素の相対アドレスを指定することを繰り返して、参照アドレスを生成する。そのため、ポスト・レジスタ・アップデート・アドレッシングの場合、インデックスレジスタとしてスカラーレジスタとベクトルレジスタの双方が指定可能である。

【0044】

デスティネーションレジスタ決定回路53は、命令コードのデスティネーションフィールドおよびリピートアmountフィールドに基づいて、デスティネーションアドレスを格納するためのデスティネーションレジスタを選択する信号（デスティネーションレジスタ選択信号）を生成する。

アドレス演算回路54は、命令パイプライン制御部51の指示に基づいて、レジスタファイル40から入力されるベースアドレスおよびインデックスアドレス（インデックスによって指定されるアドレス）から、ロード命令の対象となるメモリ10上のアドレス（ロードアドレス）を算出する。

【0045】

PR55, 56は、命令フェッチ部30から入力されるデスティネーションフィールドおよびリピートアmountフィールドのコードを一時的に記憶し、パイプライン処理において1サイクル遅延させて、デスティネーションレジスタ決定回路53に出力する。

レジスタ 57 は、命令フェッチ部 30 から入力されるベースフィールドのコードを一時的に記憶する。

【0046】

ここで、図 5 におけるアドレス演算回路 54 の構成について説明する。

図 6 は、アドレス演算回路 54 の構成例を示す図である。

図 6 において、アドレス演算回路 54 は、I レジスタ 54a と、T レジスタ 54b と、マルチプレクサ (MUX) 54c ~ 54e と、加算器 54f とを含んで構成される。

【0047】

I レジスタ 54a は、レジスタファイル 40 から入力されるインデックスアドレスを一時的に記憶する。

T レジスタ 54b は、MUX 54c を介してレジスタファイル 40 から入力されるベースアドレスを一時的に記憶する。

MUX 54c は、レジスタファイル 40 から入力されるベースアドレスと加算器 54f の出力であるアップデート・ベースアドレスとのいずれかを切り替えて、T レジスタ 54b に出力する。

【0048】

MUX 54d は、I レジスタ 54a から入力されるインデックスアドレスの上位あるいは下位のアドレスのいずれかを選択して加算器 54f に出力する。

MUX 54e は、T レジスタ 54b から入力されるベースアドレスと加算器 54f の出力であるアップデート・ベースアドレスとのいずれかを切り替えて、ベクトル命令の各繰り返しにおいて、ロード命令を実行するターゲットアドレスとして出力する。

【0049】

加算器 54f は、T レジスタ 54b から入力されるベースアドレスの値と、MUX 54d から入力されるインデックスアドレスの上位あるいは下位のデータに基づくアドレスとを加算し、アップデート・ベースアドレス (更新されたベースアドレスの値) として出力する。

なお、図 6 において、MUX 54c ~ 54e は命令パイプライン制御部 51 に

よって制御される。即ち、MUX 54 c ~ 54 e には、入力されるデータのいずれかを選択するための信号（選択指示信号）が命令パイプライン制御部 51 から入力される。

【0050】

また、アドレス演算回路 54 は、インデックス修飾アドレッシングの場合およびポスト・アップデート・アドレッシングの場合それぞれに対応して、所定の動作を行う。

次に、動作を説明する。

初めに、図 4 を参照して、ベクトルプロセッサ 1 全体の動作について説明する。

【0051】

ベクトルプロセッサ 1 において処理が行われる場合、メモリ制御部 20 を介してメモリ 10 から命令フェッチ部 30 に命令コードが読み出される。

そして、ロードユニット 50、ストアユニット 60 および演算ユニット 70 のそれぞれに、命令フェッチ部 30 から命令コードが出力される。

命令コードが入力されたロードユニット 50、ストアユニット 60 および演算ユニット 70 は、その命令コードをデコードし、それぞれのユニットに対応する命令である場合にのみ、命令を実行する。

【0052】

ここでは、図 5 を参照しつつ、命令コードがロード命令である場合について説明する。

命令コードのオペコードがロード命令を示している場合（より詳細にはprefixコードが“000”である場合）、ロードユニット 50 が動作する。

まず、ロードユニット 50 は、命令フェッチ部 30 から受け取ったベースフィールドのコードを、ベースレジスタ・リード選択信号（データの読み出し時にベースレジスタを選択する信号）としてレジスタファイル 40 に出力する。ベースレジスタ・リード選択信号は、レジスタファイル 40 内のスカラーレジスタ SR 0 ~ SR 31 のいずれかをベースレジスタとして選択するための信号である。

【0053】

そして、ベースレジスタ・リード選択信号を受け取ったレジスタファイル 40 から、ベースレジスタ・リード選択信号によって指定されたレジスタに記憶されたベースアドレスの値がロードユニット 50 に入力される。

レジスタファイル 40 から入力されたベースアドレスの値は、アドレス演算回路 54 に入力される。

【0054】

また、インデックスレジスタ決定回路 52 は、命令フェッチ部 30 から入力されたインデックスフィールドのコードおよびリピートアマウントフィールドのコードを受け取る。そして、インデックスレジスタ決定回路 52 は、命令パイプライン制御部 51 からの指示に従って、命令コードがベクトル命令であるかスカラー命令であるかを判定し、ベクトル命令である場合、インデックスレジスタ選択信号を、リピートアマウントに示される要素データ数分、レジスタファイル 40 に順次出力する。このとき、インデックスレジスタ選択信号によって指定されたレジスタがベクトルレジスタである場合、指定されたベクトルレジスタ内の各要素レジスタを特定するための所定の選択信号が出力される。

【0055】

インデックスレジスタ選択信号が入力されたレジスタファイル 40 からは、インデックスレジスタ選択信号に示されるレジスタから、インデックスアドレスの値が、アドレス演算回路 54 に順次入力される。

インデックスアドレスの値を受け取ったアドレス演算回路 54 は、ベースアドレスの値と、インデックスアドレスの値とからロードアドレスを算出し、メモリ制御部 20 に出力する。なお、アドレス演算回路 54 の動作については後述する。

【0056】

また、命令フェッチ部 30 から入力されたデスティネーションフィールドのコードおよびリピートアマウントフィールドのコードは、パイプライン処理におけるタイミングを合わせるべく、PR 55, 56 に一旦記憶された後、デスティネーションレジスタ決定回路 53 に入力される。

すると、デスティネーションレジスタ決定回路 53 は、デスティネーションレ

ジスタ選択信号を、メモリ10からロードされたデータと同期させてレジスタファイル40に出力する。

【0057】

そして、レジスタファイル40において、メモリ10からロードされたデータが、所定のデスティネーションレジスタに順次記憶される。

さらに、ポスト・レジスタ・アップデート・アドレッシングの場合、アドレス演算回路54によって出力されるアップデート・ベースアドレスをベースレジスタに書き込む。

【0058】

したがって、ベースフィールドのコードをレジスタ57に保持しておき、レジスタ57によって出力されるコードをベースレジスタ・ライト選択信号（データの書き込み時にベースレジスタを選択する信号）として用い、ベースレジスタ・ライト信号（ベースレジスタに対する書き込み指示信号）が入力されることに対応して、ベースレジスタのデータが更新される。

【0059】

次に、アドレス演算回路54の動作について説明する。

まず、インデックス修飾アドレッシングの場合について説明する。

インデックス修飾アドレッシングの場合、サイクル“1”において、ベースアドレスの値が、MUX54cを介してTレジスタ54bに格納される。

一方、インデックスアドレスの値は、Iレジスタ54aに格納される。

【0060】

次いで、サイクル“2”において、Tレジスタ54bのベースアドレスの値と、Iレジスタ54aの上位あるいは下位半分のいずれかのアドレスの値とが、加算器54fによって加算され、MUX54eを介してターゲットアドレス（ロードアドレス）として出力される。

そして、第3サイクル以後は、ベースアドレスの値がTレジスタ54bに保持された状態のまま、新たにインデックスアドレスが入力され、加算器54fによって、ベースアドレスとの加算が順次行われる。

【0061】

このとき、エクステンションフィールドのコードが入力された命令パイプライン制御部 51 が、MUX 54 d を制御することにより、図 2 に示すインデックス修飾アドレッシングの拡張仕様が実現される。

即ち、エクステンションが“000”である場合、Iレジスタ 54 a に格納されたデータ（ここでは 32 ビット）が、そのまま加算器 54 f に入力される。

【0062】

一方、エクステンションが“001”である場合、Iレジスタ 54 a の下位 16 ビットを符号拡張し、32 ビットとされたデータが加算器 54 f に入力される。

また、エクステンションが“010”である場合、Iレジスタ 54 a の上位 16 ビットを符号拡張し、32 ビットとされたデータが加算器 54 f に入力される。

【0063】

さらに、エクステンションが“011”である場合、1 サイクル毎に、Iレジスタ 54 a の下位 16 ビットと上位 16 ビットとを交互に選択し、32 ビットに符号拡張されたデータが加算器 54 f に入力される。

続いて、ポスト・レジスタ・アップデート・アドレッシングの場合について説明する。

【0064】

ポスト・レジスタ・アップデート・アドレッシングの場合、サイクル“1”において、ベースアドレスの値が、MUX 54 c を介して Tレジスタ 54 b に格納される。

一方、インデックスアドレスの値は、Iレジスタ 54 a に格納される。

次いで、サイクル“2”において、Tレジスタ 54 b のベースアドレスの値が、MUX 54 e を介して、そのままターゲットアドレス（ロードアドレス）として出力される。

【0065】

また、同時に、Tレジスタ 54 b のベースアドレスの値は、加算器 54 f にも出力され、Iレジスタ 54 a に記憶されているインデックスアドレスの値と加算

される。

次いで、加算器 54 f の加算結果は、MUX 54 c を介して、Tレジスタ 54 b に格納され、Tレジスタ 54 b がベースアドレスとして記憶する値が更新される。

【0066】

そして、第3サイクル以後は、新たにインデックスアドレスが入力され、サイクル“1”，“2”と同様に、Tレジスタ 54 b の出力をターゲットアドレスとして出力すると共に、Tレジスタ 54 b がベースアドレスとして記憶する値が更新される。

その後、実行されているロード命令の最後の要素データのターゲットアドレスを出力するのと同じタイミングで、加算器 54 f の出力がアップデート・ベースアドレスとしてレジスタファイル 40 に出力される。

【0067】

レジスタファイル 40 では、ベースレジスタ・ライト信号によって指示されたタイミングで、同時に入力されるベースレジスタ・ライト選択信号によって指定されたレジスタにアップデート・ベースアドレスを記憶する。

このとき、エクステンションフィールドのコードが入力された命令パイプライン制御部 51 が、MUX 54 d を制御することにより、図3に示すポスト・レジスタ・アップデート・アドレッシングの拡張仕様が実現される。

【0068】

即ち、エクステンションが“000”である場合、Iレジスタ 54 a に格納されたデータ（ここでは32ビット）が、そのまま加算器 54 f に入力される。

一方、エクステンションが“001”である場合、Iレジスタ 54 a の下位16ビットを符号拡張し、32ビットとされたデータが加算器 54 f に入力される。

【0069】

また、エクステンションが“010”である場合、Iレジスタ 54 a の上位16ビットを符号拡張し、32ビットとされたデータが加算器 54 f に入力される。

さらに、エクステンションが“011”である場合、1サイクル毎に、Iレジスタ54aの下位16ビットと上位16ビットとを交互に選択し、32ビットに符号拡張されたデータが加算器54fに入力される。

【0070】

また、エクステンションが“100”～“111”までである場合、エクステンションが“000”～“011”までの場合とそれぞれ同様の機能拡張を行うが、命令パイプライン制御部51がベースレジスタ・ライト信号を出力することなく、ベースレジスタの更新を行わないように制御する。

以上のように、本実施の形態に係るベクトルプロセッサ1は、インデックスによって指定されたベクトルレジスタの要素レジスタあるいはスカラーレジスタを複数の領域に分割し、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。

【0071】

したがって、1つのベクトルレジスタに、実質的に複数のインデックスベクトルを格納できることとなり、レジスタリソースを効率的に使用することが可能となる。

また、インデックスベクトルを用意する手順としては、1つのインデックスベクトル（指標ベクトル）の場合と同様であるため、プログラムのコードサイズや処理サイクルがほぼ増加することがない。

【0072】

さらに、ポスト・レジスタ・アップデート・アドレッシングの場合、インデックスレジスタとしてスカラーレジスタが指定され、そのレジスタを複数の領域に分割し、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。

そのため、インデックスレジスタとしてベクトルレジスタを用いる場合、ベクトルレジスタ全てにデータを格納した後でなければ後段の処理を開始できないのに対し、スカラーレジスタを使用できることから、スカラーレジスタにデータを格納した後、直ちに後段の処理が開始できる。

【0073】

したがって、インデックスベクトル（指標ベクトル）を用意する手順が軽減され、プログラムのコードサイズや処理サイクル数を減少させることが可能となる。

即ち、本発明によれば、間接指標ベクトル参照をより効率的に行うことが可能となる。

【0074】

なお、本実施の形態においては、アドレッシングの機能拡張のために、レジスタを上位および下位の2つの領域に分割して使用することとして説明したが、3つ以上の領域に分けて使用することも可能である。

さらに、本実施の形態においては、命令コードに含まれるエクステンションによって、その命令全体における要素レジスタのアドレスを固定的に指定する（単一のレジスタを指定したり、2つのレジスタを交互に指定する等）こととして説明したが、命令実行中に、要素レジスタの指定パターンを動的に変化させることも可能である。即ち、エクステンションのデータ長には限りがあり、複雑なパターンを指定できないことから、複雑な指定パターンを入力可能な手段を備えることにより、エクステンションによって指定可能なパターンより複雑なパターンを指定することが可能である。また、命令実行中に、要素レジスタを指定するための異なる指定パターンを入力したりすることも可能である。

（第2の実施の形態）

本実施の形態に係るベクトルプロセッサは、第1の実施の形態に係るベクトルプロセッサにおいて、図2および図3に示すアドレス修飾の拡張仕様をさらに発展させた仕様となっている。

【0075】

具体的には、命令コードにアドレッシングの機能拡張のためのコードを含め、インデックスレジスタの分割した領域を多様に指定することに加え、アドレッシングの機能拡張のための指標ベクトルを記憶する2つのレジスタ（以下、「サブインデックスレジスタA，B」と言う。）を指定対象としてさらに設けている。そして、命令コードのエクステンションのうち4ビットを使用して、アドレス修飾の方法を指定することにより、多様なアドレス修飾を可能としている。

【0076】

したがって、初めに、本実施の形態に係るベクトルプロセッサの拡張仕様について具体的に説明する。

図7は、インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。なお、図7においては、アドレス修飾の拡張仕様を説明するため、C言語に準じた記述スタイルのプログラミング例を併せて示しており、後述の図8においても同様である。

【0077】

図7において、エクステンションが“0000”である場合、インデックスとして指定されたレジスタの内容（指標ベクトル）を符号付きで解釈し、アドレス修飾を行う。

具体的には、ベースに示されたレジスタの値を常に基準とし、その値にインデックスに示されるアドレスを順次加算することにより、アドレス修飾を行う。

【0078】

また、エクステンションが“0100”である場合、インデックスとして指定されたレジスタの下位16ビットを符号付きで解釈し、アドレス修飾を行う。

さらに、エクステンションが“0110”である場合、インデックスとして指定されたレジスタの上位16ビットを符号付きで解釈し、アドレス修飾を行う。

次に、ポスト・レジスタ・アップデート・アドレッシングの拡張仕様について説明する。

【0079】

図8は、ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

図8において、エクステンションが“0000”、“0100”、“0101”、“0110”および“0111”である場合、図3における“000”、“001”、“101”、“010”、“110”とそれぞれ同様にアドレスの更新を行う。

【0080】

また、エクステンションが“1100”である場合、インデックスとして指定されたインデックスレジスタおよびサブインデックスレジスタAにおいて、下位16ビットあるいは上位16ビットのうち所定のデータを符号付きで解釈し、アドレスの更新を行う。このとき、所定のレジスタ（以下、「インデックススイッチレジスタ」と言う。）に記憶された指定パターンに基づいて、インデックスレジスタおよびサブインデックスレジスタAの上位16ビットあるいは下位16ビットのいずれかが指定される。

【0081】

具体的には、ベースに指定されたレジスタの値を所定のレジスタ（図8中のパラメータTの値を格納するレジスタ）に一旦格納し、そのレジスタの値を更新しながらアドレス修飾を行う。

また、エクステンションが“1110”の場合、インデックスとして指定されたインデックスレジスタおよびサブインデックスレジスタBにおいて、下位16ビットあるいは上位16ビットのうち所定のデータを符号付きで解釈し、アドレスの更新を行う。このとき、インデックススイッチレジスタに記憶された指定パターン（後述の「レジスタスイッチングパターン」）に基づいて、インデックスレジスタおよびサブインデックスレジスタBの上位16ビットあるいは下位16ビットのいずれかが指定される。

【0082】

ここで、エクステンションが“1100”および“1110”の場合、ベクトル命令における命令の繰り返し回数の間、インデックスとして指定された内容に応じてアドレス修飾が行われ、繰り返し回数が終了した後、ベースに指定されたレジスタの値が更新される。

したがって、複数のロード命令あるいはストア命令を用いて、繰り返しロードあるいはストアを行う場合、ベースに指定されたレジスタが自動的に更新されるので、命令発行の都度、ベースのアドレスを別の命令で更新する必要がなく、直ちに命令の実行を行うことが可能である。

【0083】

また、図8において、エクステンションが“1101”および“1111”の場合、エクステンションが“1100”および“1110”の場合とそれぞれ同様にアドレスの更新を行う。ただし、エクステンションが“1101”および“1111”の場合、命令の繰り返し回数が終了した後、ベースに指定されたレジスタの値を更新することなく命令の実行を終了する。

【0084】

次に、本実施の形態に係るベクトルプロセッサの構成を説明する。

図9は、本実施の形態に係るベクトルプロセッサ2の構成を示す図である。

図9において、ベクトルプロセッサ2の構成は、レジスタファイル140にサブインデックスレジスタ141、142とインデックススイッチレジスタ143とが含まれることを除き、第1の実施の形態におけるベクトルプロセッサ1の構成と同様である。したがって、サブインデックスレジスタ141、142およびインデックススイッチレジスタ143についてのみ説明し、他の部分については、図4において対応する部分を参照することとする。

【0085】

サブインデックスレジスタ141、142は、レジスタファイル140に含まれるスカラーレジスタあるいはベクトルレジスタ（ここでは、スカラーレジスタSR30、31とする）によって構成される。そして、サブインデックスレジスタ141、142には、所定の指標ベクトルがそれぞれ格納されている。

インデックススイッチレジスタ143は、レジスタファイル140に含まれるスカラーレジスタ（ここでは、スカラーレジスタSR29とする）によって構成され、インデックスレジスタおよびサブインデックスレジスタ141、142に含まれるいずれのデータをインデックスアドレスとして選択するかを示すデータ（以下、「レジスタスイッチングパターン」と言う。）を記憶している。

【0086】

レジスタスイッチングパターンは、具体的には、サブインデックスレジスタ141、142のいずれか一方における上位16ビットおよび下位16ビットと、後述する図11のIレジスタにおける上位16ビットおよび下位16ビットとを含む4つの領域のうち、アドレス修飾を行う要素ごとに、1つの領域を選択する

ためのデータである。

【0087】

図10は、レジスタスイッチングパターンのデータ構成を示す図である。

図10において、レジスタスイッチングパターンの上位半分には、サブインデックスレジスタ141が選択された場合のスイッチングパターンが格納され、下位半分には、サブインデックスレジスタ142が選択された場合のスイッチングパターンが格納されている。

【0088】

次に、図11は、ロードユニット150の内部構成を示すブロック図である。

図11において、ロードユニット150の内部構成は、命令パイプライン制御部151およびアドレス演算回路154の機能を除き、第1の実施の形態におけるロードユニット50の内部構成と同様である。したがって、命令パイプライン制御部151およびアドレス演算回路154についてのみ説明し、他の部分については、図5において対応する部分の説明を参照することとする。

【0089】

命令パイプライン制御部151は、ロードユニット150全体を制御するものである。また、命令パイプライン制御部151は、命令コードに含まれるエクステンションフィールドのコードに応じて、サブインデックスレジスタ141、142のいずれかのサブインデックスアドレスをアドレス演算回路154（より詳細にはMUX154g）に選択させると共に、レジスタファイル140のインデックススイッチレジスタ143における上位あるいは下位半分のレジスタスイッチングパターンを後述するJレジスタ154hに記憶させる。

【0090】

アドレス演算回路154は、命令パイプライン制御部151の指示に基づいて、レジスタファイル140から入力される、ベースアドレスおよびインデックスアドレス（インデックスフィールドによって指定されたアドレス）と、サブインデックスレジスタ141、142に記憶されたアドレス（以下、「サブインデックスアドレス」と言う。）と、インデックススイッチレジスタ143に記憶されたレジスタスイッチングパターンとから、ロード命令の対象となるメモリ110



上のアドレス（ロードアドレス）を算出する。

【0091】

続いて、図11におけるアドレス演算回路154の構成について説明する。

図12は、アドレス演算回路154の構成例を示す図である。

図12において、アドレス演算回路154の構成は、マルチプレクサ（MUX）154d、154gと、Jレジスタ154hと、シフタ154iとを除き、第1の実施の形態におけるアドレス演算回路54の構成と同様である。したがって、MUX154d、154gと、Jレジスタ154hと、シフタ154iとについての説明は、他の部分については、図6を参照において対応する部分を参照することとする。

【0092】

MUX154dは、Iレジスタ154aから入力されるインデックスアドレスの上位あるいは下位のアドレスと、後述するMUX154gから入力されるサブインデックスアドレスの上位あるいは下位のアドレスのいずれかを選択して加算器154fに出力する。このとき、MUX154dは、後述するシフタ154iから入力される指示信号に基づいて、入力されたインデックスアドレスあるいはサブインデックスアドレスの上位あるいは下位のアドレスのいずれかを選択する。

【0093】

MUX154gは、サブインデックスレジスタ141、142から入力されるサブインデックスアドレスのうち、いずれかを選択してMUX54dに出力する。このとき、MUX154gは、命令パイプライン制御部151から入力される選択指示信号に基づいて、いずれかのサブインデックスアドレスを選択する。

Jレジスタ154hは、レジスタファイル140から入力されるインデックレジスタ選択データを一時的に記憶する。

【0094】

シフタ154iは、Jレジスタ154hからレジスタスイッチングパターンを受け取り、1サイクル毎に、レジスタスイッチングパターンに基づいて、MUX54dが選択するアドレスを指示するための信号を出力する。

次に、動作を説明する。

初めに、図9を参照して、ベクトルプロセッサ2全体の動作について説明する。

【0095】

ベクトルプロセッサ2において処理が行われる場合、メモリ制御部120を介してメモリ110から命令フェッチ部130に命令コードが読み出される。

そして、ロードユニット150、ストアユニット160および演算ユニット170のそれぞれに、命令フェッチ部130から命令コードが出力される。

命令コードが入力されたロードユニット150、ストアユニット160および演算ユニット170は、その命令コードをデコードし、それぞれのユニットに対応する命令である場合にのみ、命令を実行する。

【0096】

ここでは、図11を参照しつつ、命令コードがロード命令である場合について説明する。

命令コードのオペコードがロード命令を示している場合（より詳細にはprefixコードが“000”である場合）、ロードユニット150が動作する。

まず、ロードユニット150は、命令フェッチ部130から受け取ったベースフィールドのコードを、ベースレジスタ・リード選択信号（データの読み出し時にベースレジスタを選択する信号）としてレジスタファイル140に出力する。

【0097】

そして、ベースレジスタ・リード選択信号を受け取ったレジスタファイル140から、ベースレジスタ・リード選択信号によって指定されたレジスタに記憶されたベースアドレスの値がロードユニット150に入力される。

レジスタファイル140から入力されたベースアドレスの値は、アドレス演算回路154に入力される。

【0098】

また、インデックスレジスタ決定回路152は、命令フェッチ部130から入力されたインデックスフィールドのコードおよびリピートアmountフィールドのコードを受け取る。そして、インデックスレジスタ決定回路152は、命令パ

イプライン制御部 151 からの指示に従って、命令コードがベクトル命令であるかスカラー命令であるかを判定し、ベクトル命令である場合、インデックスレジスタ選択信号を、リピートアマウントに示される要素データ数分、レジスタファイル 140 に順次出力する。このとき、インデックスレジスタ選択信号によって指定されたレジスタがベクトルレジスタである場合、指定されたベクトルレジスタ内の各要素レジスタを特定するための所定の選択信号が出力される。

【0099】

インデックスレジスタ選択信号が入力されたレジスタファイル 140 からは、インデックスレジスタ選択信号に示されるレジスタから、インデックスアドレスの値が、アドレス演算回路 154 に順次入力される。

また、命令パイプライン制御部 151 は、レジスタファイル 140 に、サブインデックスアドレス 141, 142 およびインデックススイッチレジスタ 143 のアドレスを出力する。そして、インデックスアドレスの値がアドレス演算回路 154 に入力されるタイミングに合わせて、サブインデックスアドレスの値も、アドレス演算回路 154 に順次入力される。

【0100】

インデックスアドレスおよびサブインデックスアドレスの値を受け取ったアドレス演算回路 154 は、レジスタスイッチングパターンに基づいて、インデックスアドレスおよびサブインデックスアドレスのうち、所定の領域を選択し、ベースアドレスの値と、選択した領域に記憶されているアドレスの値とからロードアドレスを算出し、メモリ制御部 120 に出力する。なお、アドレス演算回路 154 の動作については後述する。

【0101】

また、命令フェッチ部 130 から入力されたデスティネーションフィールドのコードおよびリピートアマウントフィールドのコードは、パイプライン処理におけるタイミングを合わせるべく、PR 155, 156 に一旦記憶された後、デスティネーションレジスタ決定回路 153 に入力される。

すると、デスティネーションレジスタ決定回路 153 は、デスティネーションレジスタ選択信号を、メモリ 110 からロードされたデータと同期させてレジス

タファイル 140 に出力する。

【0102】

そして、レジスタファイル 140 において、メモリ 110 からロードされたデータが、所定のデスティネーションレジスタに順次記憶される。

さらに、ポスト・レジスタ・アップデート・アドレッシングの場合、アドレス演算回路 154 によって出力されるアップデート・ベースアドレスをベースレジスタに書き込む。

【0103】

したがって、ベースフィールドのコードをレジスタ 157 に保持しておき、レジスタ 157 によって出力されるコードをベースレジスタ・ライト選択信号（データの書き込み時にベースレジスタを選択する信号）として用い、ベースレジスタ・ライト信号（ベースレジスタに対する書き込み指示信号）が入力されることに対応して、ベースレジスタのデータが更新される。

【0104】

次に、アドレス演算回路 154 の動作について説明する。

まず、インデックス修飾アドレッシングの場合について説明する。

インデックス修飾アドレッシングの場合、サイクル“1”において、ベースアドレスの値が、MUX 154 c を介して T レジスタ 154 b に格納される。

一方、インデックスアドレスの値は、I レジスタ 154 a に格納され、レジスタスイッチングパターンの上位あるいは下位半分のデータが、J レジスタ 154 h に格納される。

【0105】

次いで、サイクル“2”において、J レジスタ 154 h に格納されているレジスタスイッチングパターン（上位あるいは下位半分）における最下位 2 ビットがシフタ 154 i によって MUX 154 d に出力される。そして、MUX 154 d は、シフタ 154 i から入力された 2 ビットのデータに基づいて、MUX 154 g を介して入力されたサブインデックスアドレスの上位あるいは下位いずれかの 16 ビットのデータを選択し、加算器 154 f に出力する。また、T レジスタ 154 b のベースアドレスの値と、MUX 154 d から出力されたアドレスの値と

が、加算器 154 f によって加算され、MUX 154 e を介してターゲットアドレス（ロードアドレス）として出力される。

【0106】

そして、第3サイクル以後は、ベースアドレスの値がTレジスタ 154 b に保持された状態のまま、シフタ 154 i によって、J レジスタ 154 h に格納されたレジスタスイッチングパターン（上位あるいは下位半分）において、下位から2ビットずつがMUX 154 d に順次出力され、加算器 154 f によって、MUX 154 d から出力されるアドレスが、ベースアドレスと順次加算される。

【0107】

このとき、エクステンションフィールドのコードが入力された命令パイプライン制御部 151 が、MUX 154 g を制御すると共に、エクステンションフィールドのコードに応じて、レジスタスイッチングパターンの上位あるいは下位半分のデータのいずれかを選択してJ レジスタ 154 h に記憶させることにより、図7に示すインデックス修飾アドレッシングの拡張仕様が実現される。

【0108】

即ち、エクステンションが“0000”である場合、I レジスタ 154 a に格納されたデータ（ここでは32ビット）がMUX 154 d によって選択され、そのまま加算器 154 f に入力される。

一方、エクステンションが“0100”である場合、I レジスタ 154 a の下位16ビットを符号拡張し、32ビットとされたデータがMUX 154 d によって選択され、加算器 154 f に入力される。

【0109】

また、エクステンションが“0110”である場合、I レジスタ 154 d の上位16ビットを符号拡張し、32ビットとされたデータがMUX 154 d によって選択され、加算器 154 f に入力される。

続いて、ポスト・レジスタ・アップデート・アドレッシングの場合について説明する。

【0110】

ポスト・レジスタ・アップデート・アドレッシングの場合、サイクル“1”に

において、ベースアドレスの値が、MUX 154 c を介して Tレジスタ 154 b に格納される。

一方、インデックスアドレスの値は、Iレジスタ 154 a に格納され、レジスタスイッチングパターンの上位あるいは下位半分のデータが、命令パイプライン制御部 151 の指示に従って J レジスタ 154 h に格納される。

【0111】

次いで、サイクル“2”において、Tレジスタ 154 b のベースアドレスの値が、MUX 154 e を介して、そのままターゲットアドレス（ロードアドレス）として出力される。

また、同時に、Tレジスタ 154 b のベースアドレスの値は、加算器 154 f にも出力され、MUX 154 d から出力されたアドレスの値と加算される。

【0112】

次いで、加算器 154 f の加算結果は、MUX 154 c を介して、Tレジスタ 154 b に格納され、Tレジスタ 154 b がベースアドレスとして記憶する値が更新される。

そして、第3サイクル以後は、シフタ 154 i によって、J レジスタ 154 h に格納されたレジスタスイッチングパターン（上位あるいは下位半分）において、下位から2ビットずつがMUX 154 d に順次出力され、サイクル“1”，“2”と同様に、Tレジスタ 154 b の出力をターゲットアドレスとして出力すると共に、Tレジスタ 154 b がベースアドレスとして記憶する値が更新される。

【0113】

その後、実行されているロード命令の最後の要素データのターゲットアドレスを出力するのと同じタイミングで、加算器 154 f の出力がアップデート・ベースアドレスとしてレジスタファイル 140 に出力される。

レジスタファイル 140 では、ベースレジスタ・ライト信号によって指示されたタイミングで、同時に入力されるベースレジスタ・ライト選択信号によって指定されたレジスタにアップデート・ベースアドレスを記憶する。

【0114】

このとき、エクステンションフィールドのコードが入力された命令パイプライン

ン制御部 151 が、MUX 154 g を制御すると共に、エクステンションフィールドのコードに応じて、レジスタスイッチングパターンの上位あるいは下位半分のデータのいずれかを選択して J レジスタ 154 h に記憶させることにより、図 8 に示すポスト・レジスタ・アップデート・アドレッシングの拡張仕様が実現される。

【0115】

即ち、エクステンションが“0000”である場合、I レジスタ 154 a に格納されたデータ（ここでは 32 ビット）が MUX 154 d によって選択され、そのまま加算器 154 f に入力される。

一方、エクステンションが“0100”である場合、I レジスタ 154 a に格納されたデータの下位 16 ビットが MUX 154 d によって選択され、符号拡張して 32 ビットとされたデータが加算器 154 f に入力される。

【0116】

また、エクステンションが“0110”である場合、I レジスタ 154 a に格納されたデータの上位 16 ビットが MUX 154 d によって選択され、符号拡張して 32 ビットとされたデータが加算器 154 f に入力される。

また、エクステンションが“1100”である場合、I レジスタ 154 a に格納されたデータおよびサブインデックスレジスタ 141 に格納されたデータそれぞれにおいて、下位 16 ビットあるいは上位 16 ビットのうち、レジスタスイッチングパターンに示されるデータが MUX 154 d によって順次選択され、符号拡張して 32 ビットとされたデータが加算器 154 f に入力される。

【0117】

また、エクステンションが“1110”である場合、I レジスタ 154 a に格納されたデータおよびサブインデックスレジスタ 142 に格納されたデータそれぞれにおいて、下位 16 ビットあるいは上位 16 ビットのうち、レジスタスイッチングパターンに示されるデータが MUX 154 d によって順次選択され、符号拡張して 32 ビットとされたデータが加算器 154 f に入力される。

【0118】

また、エクステンションが“0101”、“0111”、“1101”および

“1111”である場合、エクステンションが“0100”、“0110”、“1100”および“1110”の場合とそれぞれ同様の機能拡張を行うが、命令パイプライン制御部151がベースレジスタ・ライト信号を出力することなく、ベースレジスタの更新を行わないように制御する。

【0119】

以上のように、本実施の形態に係るベクトルプロセッサ2は、インデックスによって指定されたベクトルレジスタの要素レジスタあるいはスカラーレジスタを複数の領域に分割すると共に、指標ベクトルを格納した他のレジスタ（サブインデックスレジスタ141, 142）を用意し、このレジスタも同様に、複数の領域に分割して使用する。そして、インデックスによって指定されたレジスタおよび用意した他のレジスタにおいて、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。

【0120】

したがって、少数のレジスタに、実質的に、より多くのインデックスベクトルを格納できることとなり、レジスタリソースを効率的に使用することが可能となる。

また、ベクトルプロセッサ2は、インデックススイッチレジスタ143を備えている。そして、インデックススイッチレジスタ143に記憶されたレジスタスイッチングパターンによって、アドレス修飾を行う要素毎に、インデックスレジスタおよびサブインデックスレジスタ141, 142の上位あるいは下位の領域のいずれかを任意に指定することが可能である。

【0121】

したがって、1つの命令を実行する際に、インデックスレジスタおよびサブインデックスレジスタ141, 142における領域のいずれを選択するかを動的に変化させることができ、領域を指定することによって得られるインデックスベクトルを、より多様なものとすることができる。

即ち、本発明によれば、間接指標ベクトル参照をより効率的に行うことが可能となる。

【0122】

なお、本実施の形態においては、レジスタスイッチングパターンをインデックススイッチレジスタ143に格納しておくこととして説明したが、レジスタスイッチングパターンを命令コードのインデックスフィールドに含めることとしてもよい。

図13は、レジスタスイッチングパターンを命令コードのインデックスフィールドに含める場合のデータ構成を示す図である。

【0123】

図13において、インデックスフィールドには、レジスタスイッチングパターンと、第1および第2のインデックスが格納されたそれぞれの領域（例えば、インデックスレジスタの上位および下位16ビット）を示すアドレスとが含まれている。

そして、アドレス修飾を行う8個の要素それぞれについて、2つのインデックスのいずれによって、アドレス修飾を行うかが任意に指定されている。

【0124】

また、この場合、図12のアドレス演算回路154の構成は、以下のようになる。

図14は、レジスタスイッチングパターンを命令コードのインデックスフィールドに含める場合のアドレス演算回路154の構成を示す図である。

図14においては、レジスタスイッチングパターンが命令コードに含まれていることから、インデックススイッチレジスタ143に相当するレジスタが設けられておらず、レジスタファイル140を介して、Iレジスタ154aにレジスタスイッチングパターンと、第1および第2のインデックスが格納されたそれぞれの領域のアドレスとが入力される。

【0125】

そして、入力されたレジスタスイッチングパターンは、シフタ154iに入力され、アドレス修飾を行う要素毎に、Iレジスタ154aに記憶された2つのインデックスのいずれかを選択するための信号をMUX154dに出力する。

このような構成とすることにより、インデックススイッチレジスタ143を設

けることなく、領域を指定することによって得られるインデックスベクトルを、多様なものとすることが可能となる。

【図面の簡単な説明】

【図 1】 ロード命令あるいはストア命令のフォーマットを示す図である。

【図 2】 インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図 3】 ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図 4】 本実施の形態に係るベクトルプロセッサ 1 の構成を示す図である。

【図 5】 ロードユニット 50 の内部構成を示すブロック図である。

【図 6】 アドレス演算回路 54 の構成例を示す図である。

【図 7】 インデックス修飾アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図 8】 ポスト・レジスタ・アップデート・アドレッシングの場合において、エクステンションによって示されるコードと、アドレス修飾の拡張仕様との対応関係を示す図である。

【図 9】 本実施の形態に係るベクトルプロセッサ 2 の構成を示す図である。

【図 10】 レジスタスイッチングパターンのデータ構成を示す図である。

【図 11】 ロードユニット 150 の内部構成を示すブロック図である。

【図 12】 アドレス演算回路 154 の構成例を示す図である。

【図 13】 レジスタスイッチングパターンを命令コードのインデックスフィールドに含める場合のデータ構成を示す図である。

【図 14】 レジスタスイッチングパターンを命令コードのインデックスフィールドに含める場合のアドレス演算回路 154 の構成を示す図である。

【符号の説明】

1, 2 ベクトルプロセッサ, 10, 110 メモリ, 20, 120 メモリ制御部, 30, 130 命令フェッチ部, 40, 140 レジスタファイル, 50, 150 ロードユニット, 51, 151 命令パイプライン制御部, 52, 152 インデックスレジスタ決定回路, 53, 153 デスティネーションレジスタ決定回路, 54, 154 アドレス演算回路, 54 a, 154 a Iレジスタ, 54 b, 154 b Tレジスタ, 54 c~54 e, 154 c~154 e, 154 g MUX (マルチプレクサ), 54 f, 154 f 加算器, 145 h Jレジスタ, 154 i シフタ, 55, 56, 155, 156 PR (パイプラインレジスタ), 57, 157 レジスタ, 60, 160 ストアユニット, 70, 170 演算ユニット

【書類名】

図面

【図 1】

(a) LS0タイプ(ベース相対アドレッシング)

opcode	dst	base	offset
--------	-----	------	--------

(b) LS1タイプ(ポスト・オフセット・アップデート・アドレッシング)

opcode	dst	base	rptamt	offset
--------	-----	------	--------	--------

(c) LS2タイプ(インデックス修飾／ポスト・レジスタ・アップデート・アドレッシング)

opcode	dst	base	rptamt	extension	index
--------	-----	------	--------	-----------	-------

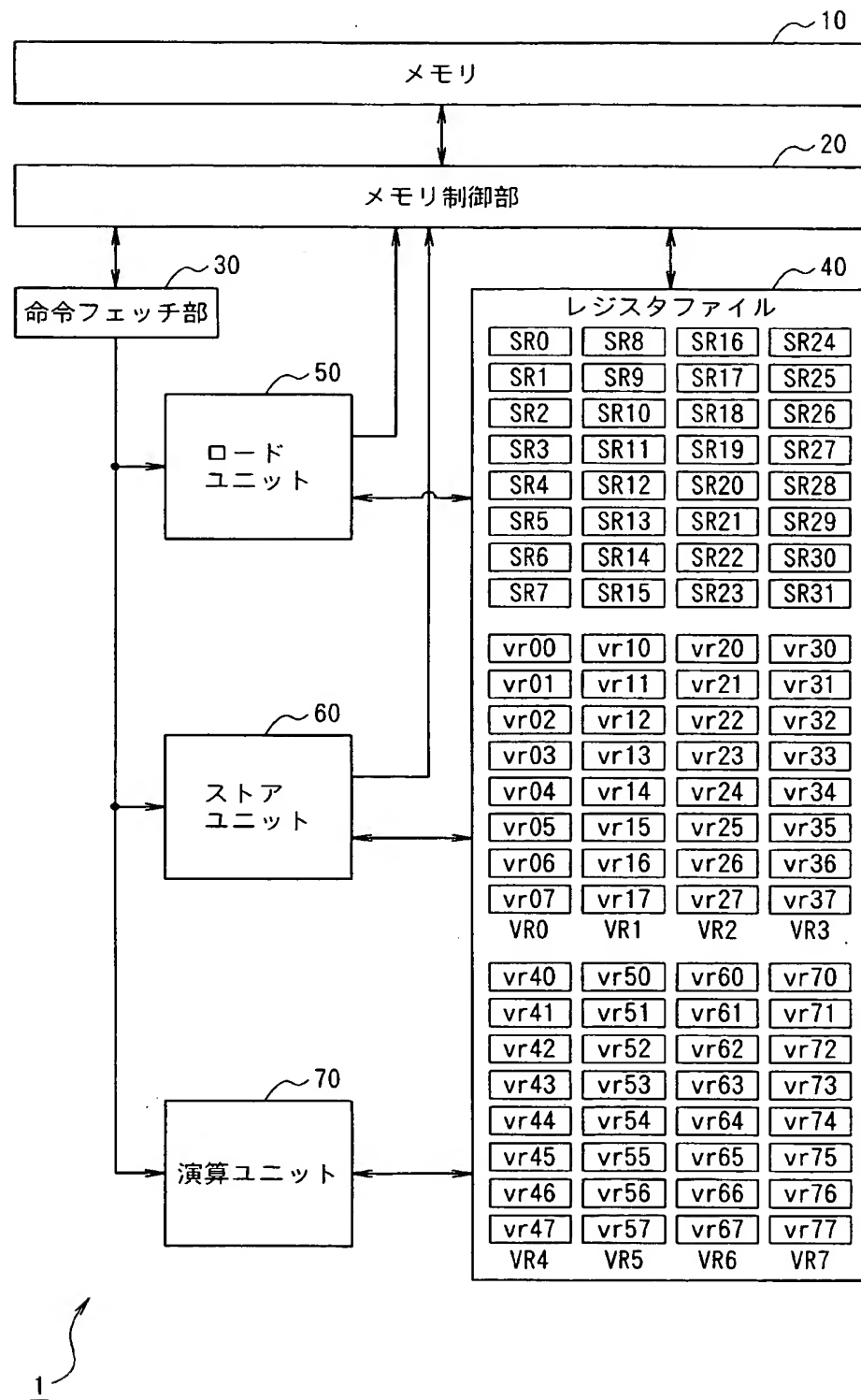
【図 2】

extension	アドレス修飾の方法
000	<p>インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレス修飾を行う。</p> <pre> for(t=0; t < rptamt; t++) targetAdrs = base.uw + index[t].h[LOWER]; </pre>
001	<p>インデックスとして指定されたレジスタの下位 16 ビット (index[t].h[LOWER]) を符号付きで解釈し、アドレス修飾を行う。</p> <pre> for(t=0; t < rptamt; t++) targetAdrs = base.uw + index[t].h[LOWER]; </pre>
010	<p>インデックスとして指定されたレジスタの上位 16 ビット (index[t].h[UPPER]) を符号付きで解釈し、アドレス修飾を行う。</p> <pre> for(t=0; t < rptamt; t++) targetAdrs = base.uw + index[t].h[UPPER]; </pre>
011	<p>インデックスとして指定されたレジスタの下位 16 ビット (index[t].h[LOWER]) および上位 16 ビット (index[t].h[UPPER]) をそれぞれ符号付きで解釈し、交互にアドレス修飾を行う。</p> <pre> for(t=0; t < rptamt; t++) if ((t & 1) == 0) targetAdrs = base.uw + index[t].h[LOWER]; else targetAdrs = base.uw + index[t].h[UPPER]; </pre>

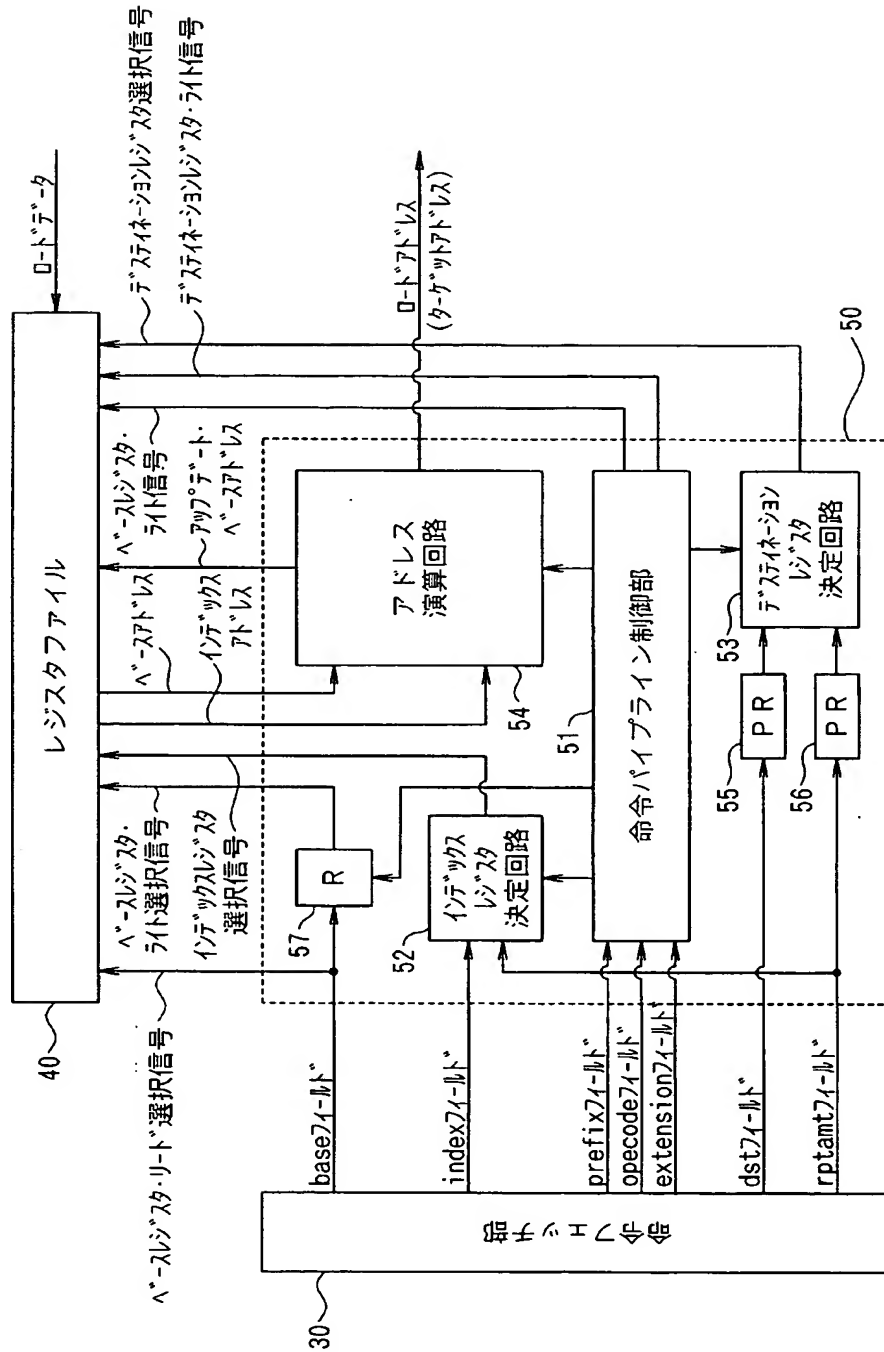
【図 3】

extension	アドレス修飾の方法
000	<p>インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレスの更新を行う。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].w; } base.uw = T; </pre>
001	<p>インデックスとして指定されたレジスタの下位16ビットのindex[t].h[LOWER]を符号付きで解釈し、アドレスの更新を行う。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[LOWER]; } base.uw = T; </pre>
010	<p>インデックスとして指定されたレジスタの上位16ビットのindex[t].h[UPPER]を符号付きで解釈し、アドレスの更新を行う。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[UPPER]; } base.uw = T; </pre>
011	<p>インデックスとして指定されたレジスタの下位16ビットのindex[t].h[LOWER]と上位16ビットのindex[t].h[UPPER]をそれぞれ符号付きで解釈し、交互にアドレスの更新を行う。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; if ((t & 1) == 0) T += index[t].h[LOWER]; else T += index[t].h[UPPER]; } base.uw = T; </pre>
100	<p>インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレスの更新を行う。ベース・レジスタの更新は行わない。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].w; } </pre>
101	<p>インデックスとして指定されたレジスタの下位16ビットのindex[t].h[LOWER]を符号付きで解釈し、アドレスの更新を行う。ベース・レジスタの更新は行わない。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[LOWER]; } </pre>
110	<p>インデックスとして指定されたレジスタの上位16ビットのindex[t].h[UPPER]を符号付きで解釈し、アドレスの更新を行う。ベース・レジスタの更新は行わない。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[UPPER]; } </pre>
111	<p>インデックスとして指定されたレジスタの下位16ビットのindex[t].h[LOWER]と上位16ビットのindex[t].h[UPPER]をそれぞれ符号付きで解釈し、交互にアドレスの更新を行う。ベース・レジスタの更新は行わない。</p> <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; if ((t & 1) == 0) T += index[t].h[LOWER]; else T += index[t].h[UPPER]; } </pre>

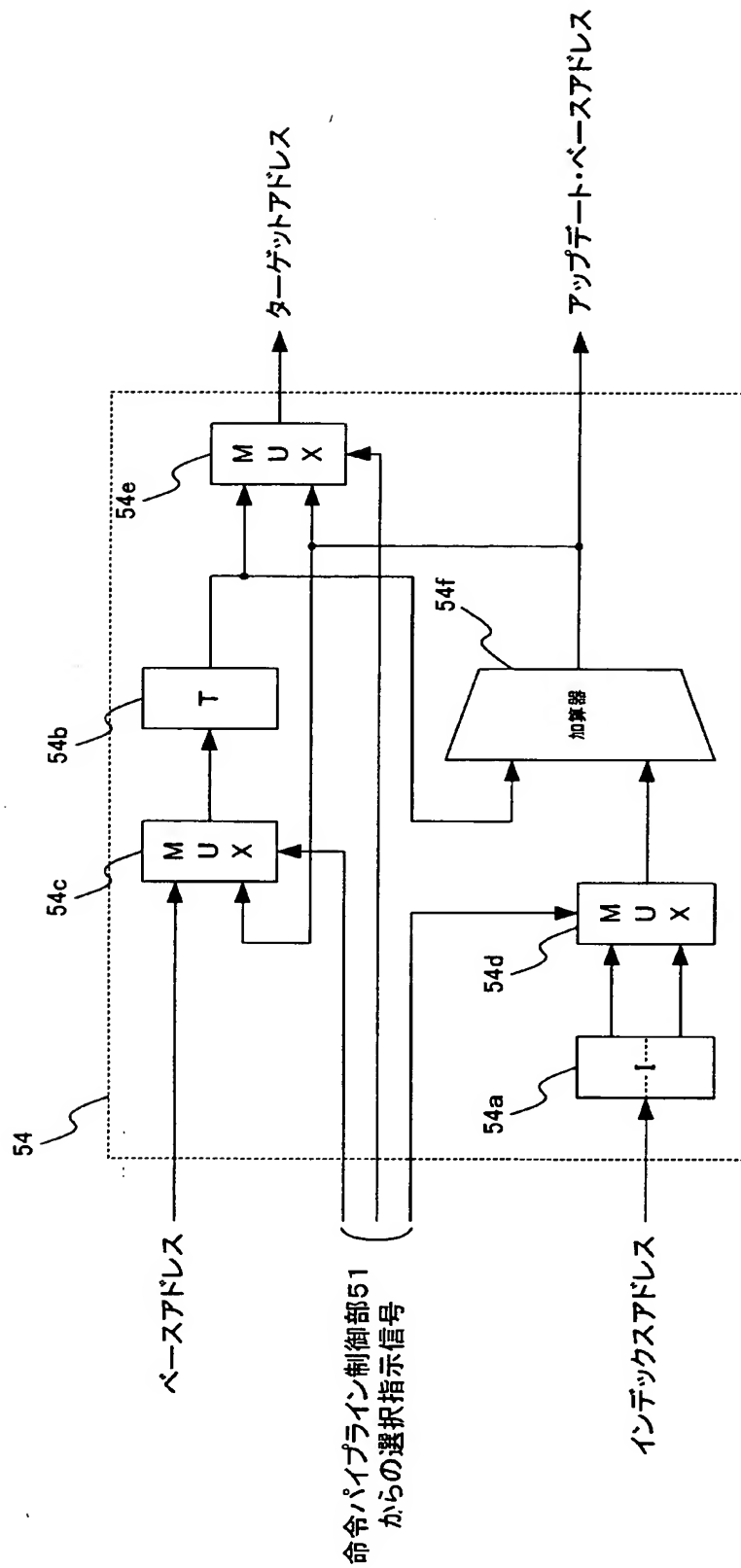
【図 4】



【図5】



【図 6】



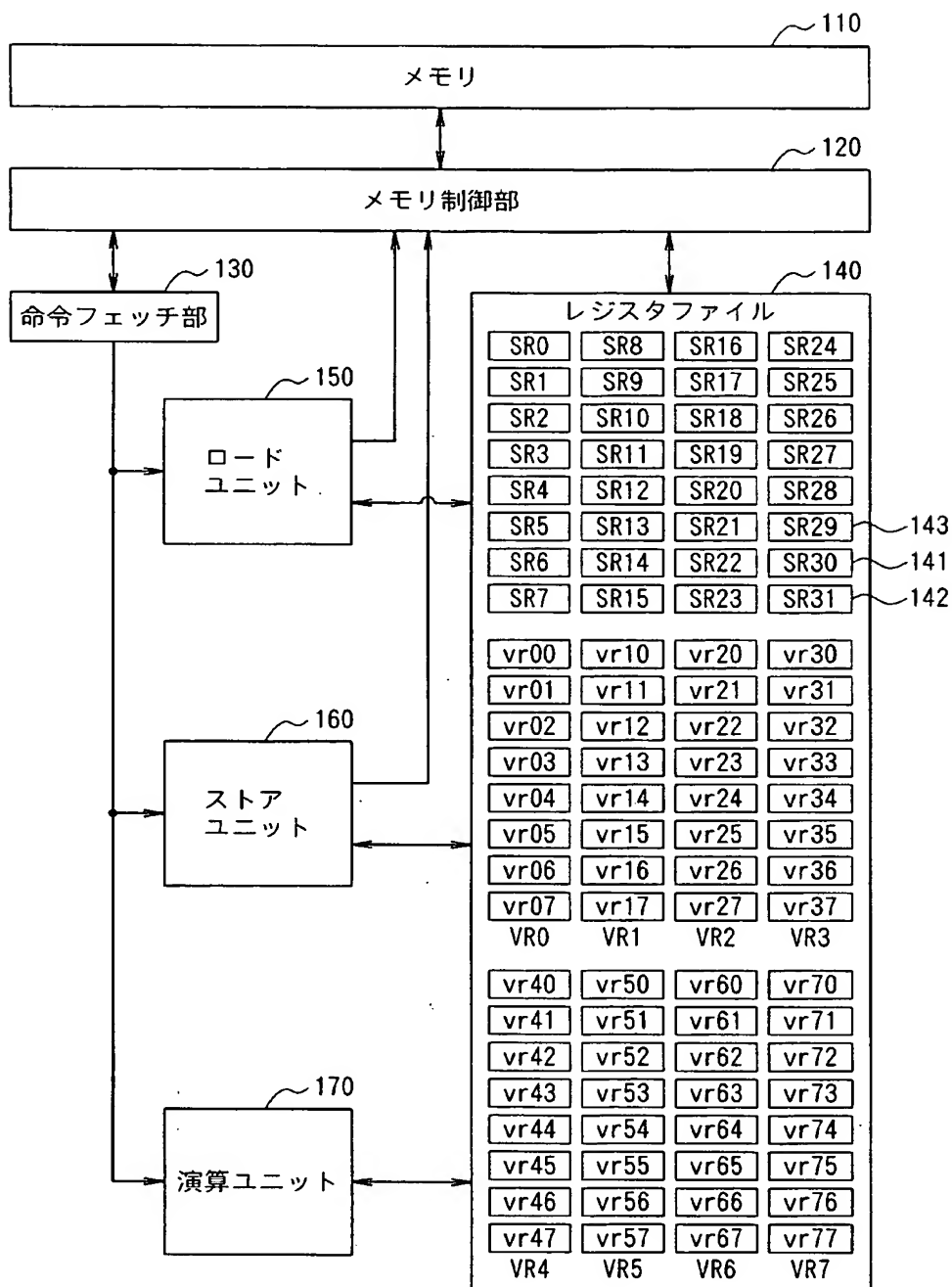
【図 7】

extension	アドレス修飾の方法
0000	インデックスとして指定されたレジスタの内容を符号付きで解釈し、アドレス修飾を行う。 for(t=0; t < rptamt; t++) targetAdrs = base.uw + index[t].w; }
0100	インデックスとして指定されたレジスタの下位 16 ビット (index[t].h[LOWER]) を符号付きで解釈し、アドレス修飾を行う。 for(t=0; t < rptamt; t++) targetAdrs = base.uw + index[t].h[LOWER]; }
0110	インデックスとして指定されたレジスタの上位 16 ビット (index[t].h[UPPER]) を符号付きで解釈し、アドレス修飾を行う。 for(t=0; t < rptamt; t++) targetAdrs = base.uw + index[t].h[UPPER]; }

【図 8】

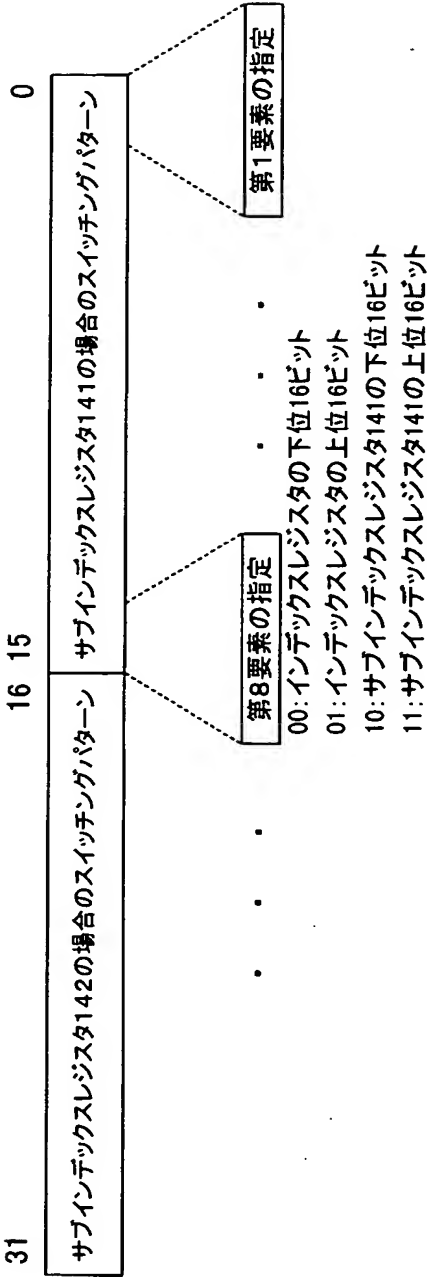
extension	アドレス修飾の方法
0000	インデックスとして指定したレジスタの内容を符号付きで解釈し、アドレスの更新を行う。 <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].w; } base.uw = T; </pre>
0100	インデックスとして指定したレジスタの下位16ビットのindex[t].h[LOWER]を符号付きで解釈し、アドレスの更新を行う。 <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[LOWER]; } base.uw = T; </pre>
0101	同上、但し、ベース・レジスタの更新を行わない。 <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[LOWER]; } </pre>
0110	インデックスとして指定したレジスタの上位16ビットのindex[t].h[UPPER]を符号付きで解釈し、アドレスの更新を行う。 <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[UPPER]; } base.uw = T; </pre>
0111	同上、但し、ベース・レジスタの更新を行わない。 <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; T += index[t].h[UPPER]; } </pre>
1100	インデックスとして指定したレジスタの下位16ビットのindex.h[LOWER]と上位16ビットのindex.h[UPPER]およびサブインデックスレジスタAの下位16ビットのSR30.h[LOWER]および上位16ビットのSR30.h[UPPER]のうち所定データを符号付きで解釈し、アドレスの更新を行う。 <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; switch((ISW.h[LOWER] >> (2 * t)) & 0x3){ case 0: T += index.h[LOWER]; break; case 1: T += index.h[UPPER]; break; case 2: T += SR30.h[LOWER]; break; case 3: T += SR30.h[UPPER]; break; } } base.uw = T; </pre>
1101	同上、但し、ベース・レジスタの更新を行わない。 <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; switch((ISW.h[UPPER] >> (2 * t)) & 0x3){ case 0: T += index.h[LOWER]; break; case 1: T += index.h[UPPER]; break; case 2: T += SR30.h[LOWER]; break; case 3: T += SR30.h[UPPER]; break; } } </pre>
1110	インデックスとして指定したレジスタの下位16ビットのindex.h[LOWER]と上位16ビットのindex.h[UPPER]およびサブインデックスレジスタBの下位16ビットのSR31.h[LOWER]および上位16ビットのSR31.h[UPPER]のうち所定データを符号付きで解釈し、アドレスの更新を行う。 <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; switch((ISW.h[LOWER] >> (2 * t)) & 0x3){ case 0: T += index.h[LOWER]; break; case 1: T += index.h[UPPER]; break; case 2: T += SR31.h[LOWER]; break; case 3: T += SR31.h[UPPER]; break; } } base.uw = T; </pre>
1111	同上、但し、ベース・レジスタの更新を行わない。 <pre> T = base.uw; for(t=0; t < rptamt; t++){ targetAdrs = T; switch((ISW.h[UPPER] >> (2 * t)) & 0x3){ case 0: T += index.h[LOWER]; break; case 1: T += index.h[UPPER]; break; case 2: T += SR31.h[LOWER]; break; case 3: T += SR31.h[UPPER]; break; } } </pre>

【図 9】

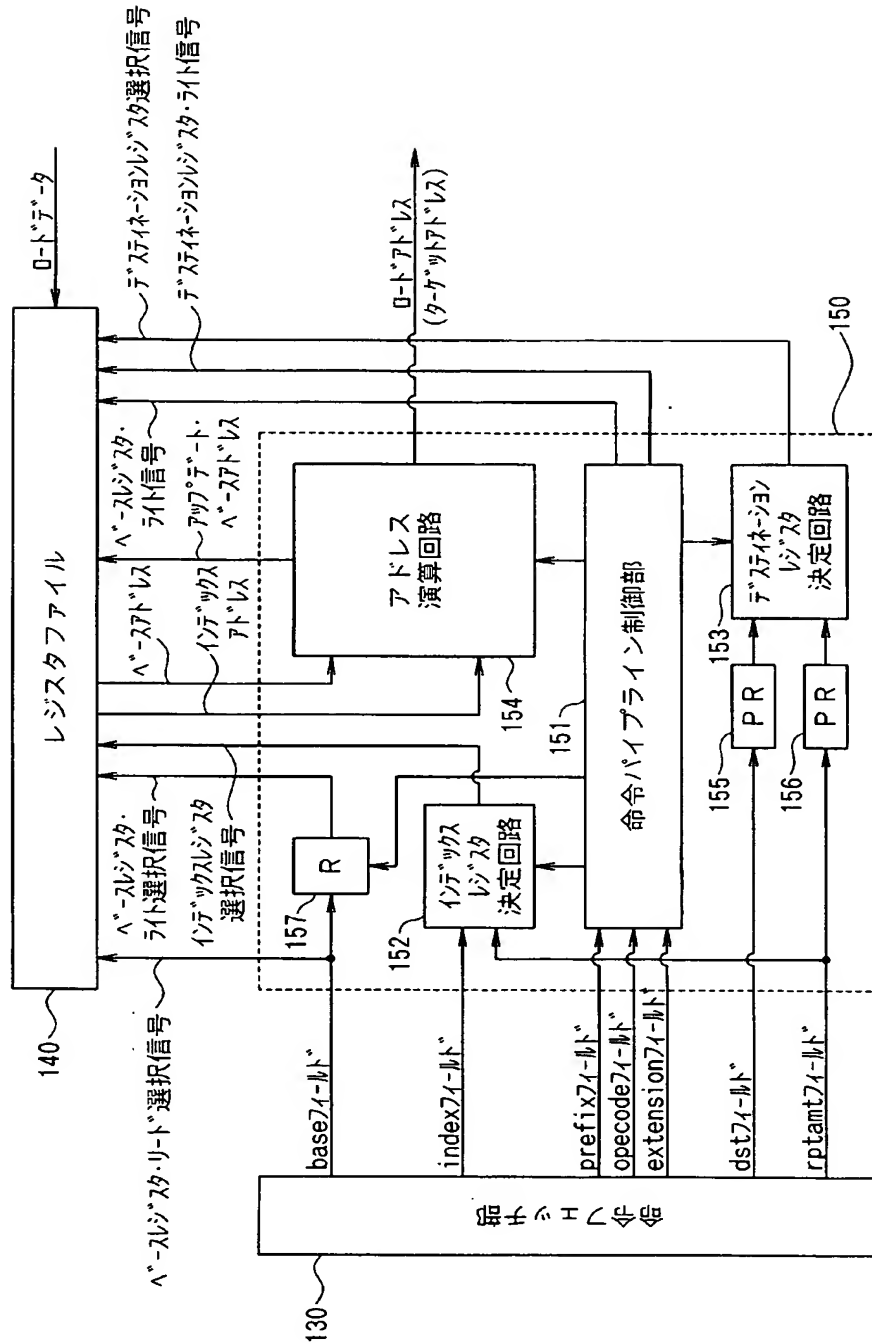


2

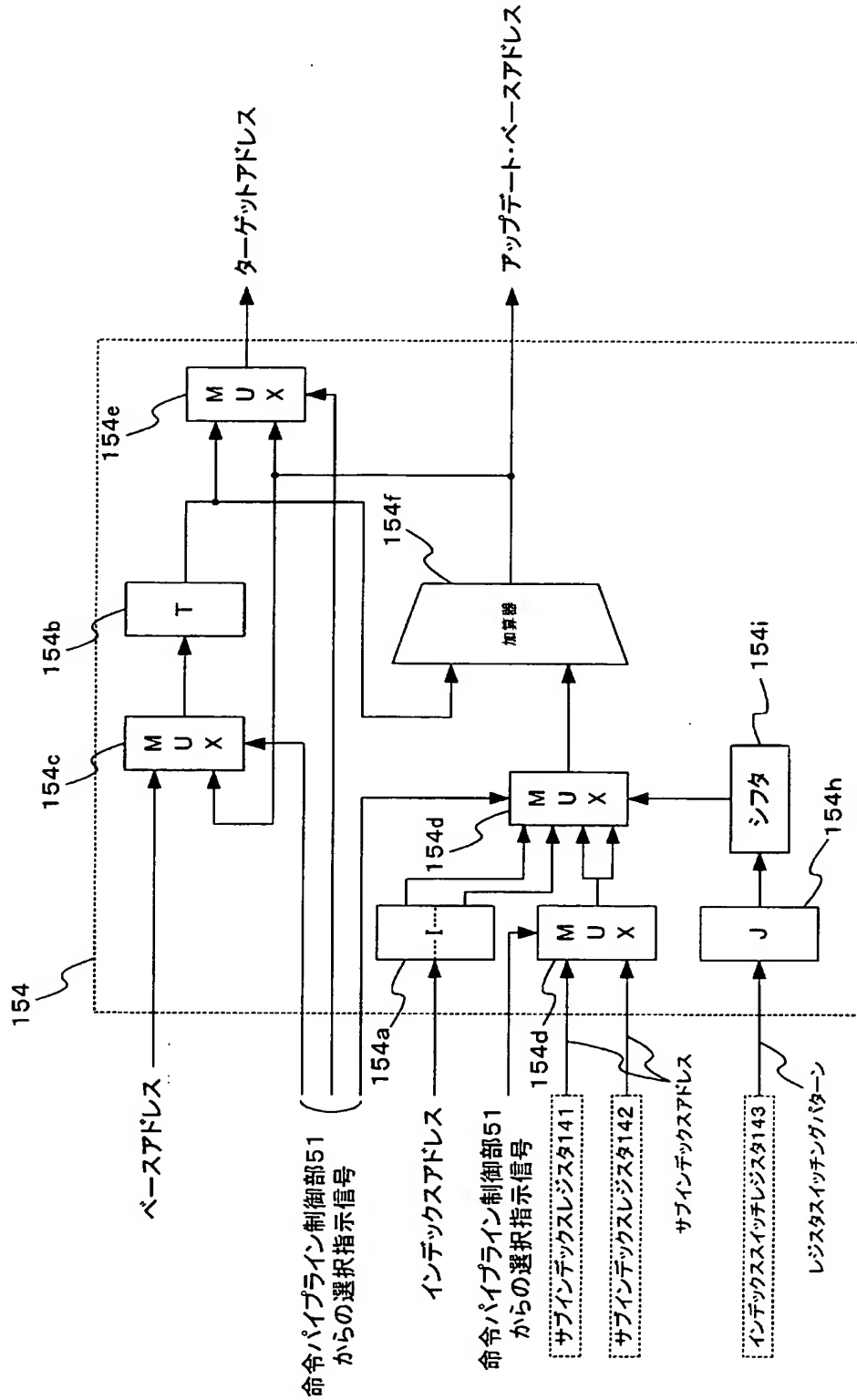
【図 10】



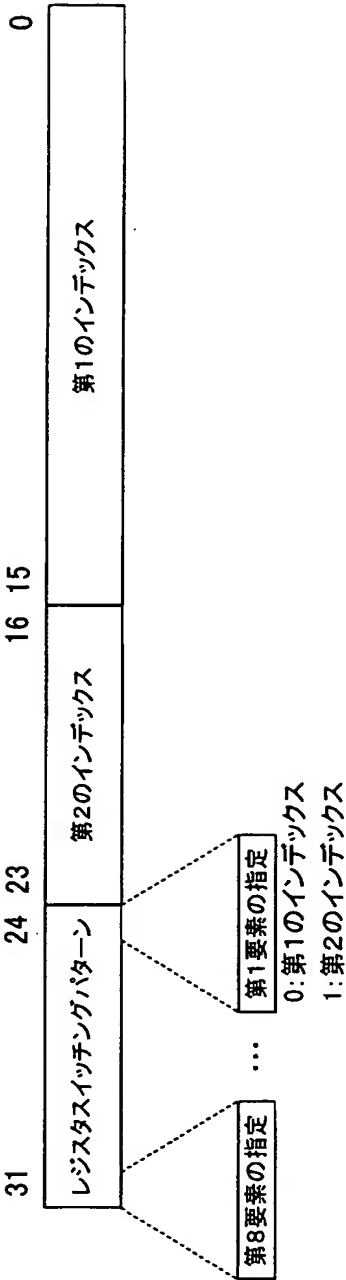
【図 11】



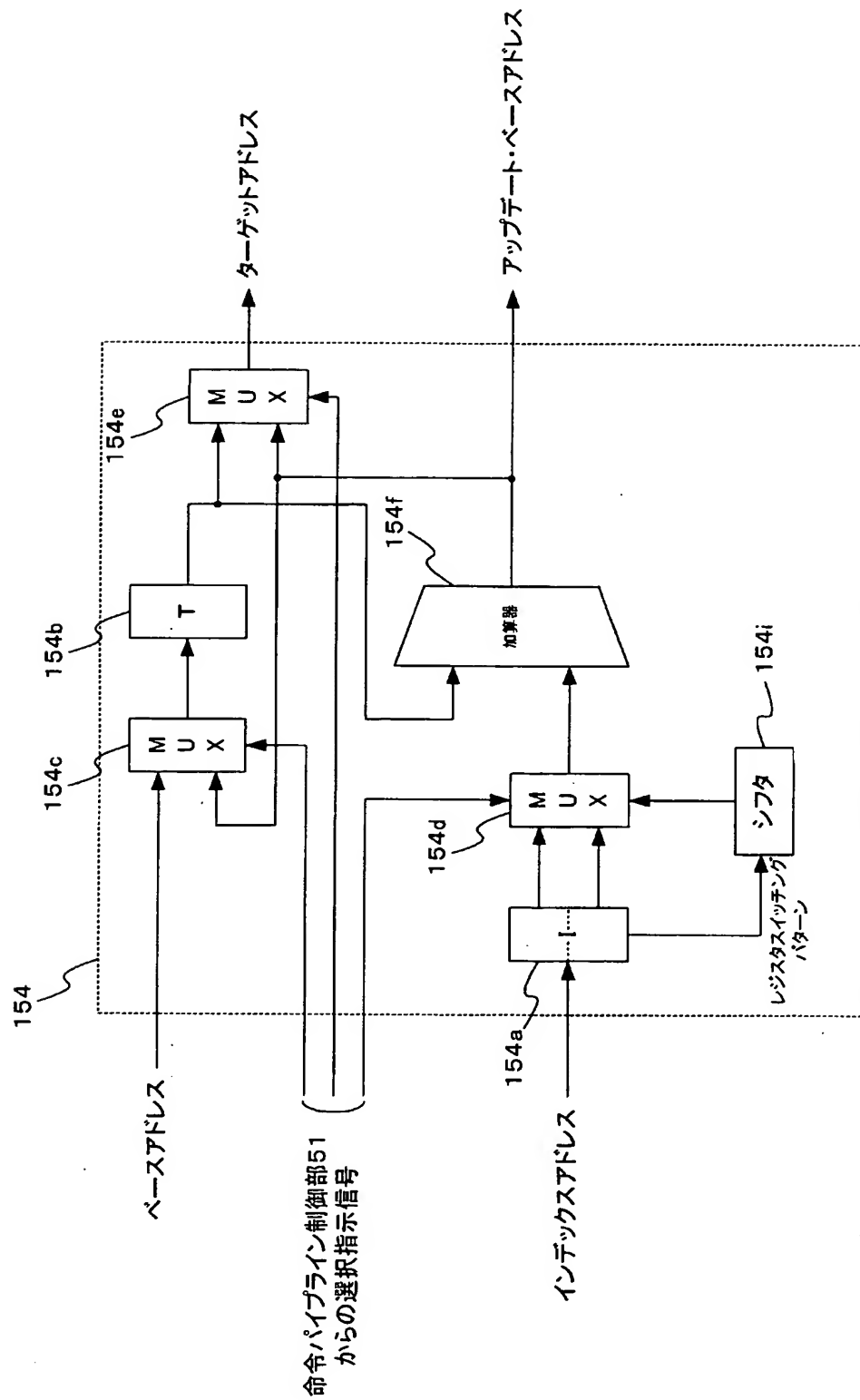
【図 12】



【図 13】



【図 14】



【書類名】 要約書

【要約】

【課題】 間接指標ベクトル参照をより効率的に行うこと。

【解決手段】 インデックスによって指定されたベクトルレジスタの要素レジスタあるいはスカラーレジスタを複数の領域に分割し、分割した領域のいずれかを選択することによって、所定のインデックスベクトル（指標ベクトル）を取得することとしている。したがって、1つのベクトルレジスタに、実質的に複数のインデックスベクトルを格納できることとなり、ベクトルレジスタを効率的に使用することが可能となる。また、インデックスベクトルを用意する手順としては、1つのインデックスベクトルの場合と同様であるため、プログラムのコードサイズや処理サイクルがほぼ増加することがない。即ち、本発明によれば、間接指標ベクトル参照をより効率的に行うことが可能となる。

【選択図】 図2

認定・付加情報

特許出願の番号	特願 2003-203856
受付番号	50301264043
書類名	特許願
担当官	第七担当上席 0096
作成日	平成15年 8月 4日

<認定情報・付加情報>

【提出日】 平成15年 7月30日

【特許出願人】

【識別番号】 000002369

【住所又は居所】 東京都新宿区西新宿2丁目4番1号

【氏名又は名称】 セイコーエプソン株式会社

【代理人】 申請人

【識別番号】 100095728

【住所又は居所】 長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社 知的財産本部内

【氏名又は名称】 上柳 雅誉

【選任した代理人】

【識別番号】 100107261

【住所又は居所】 長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社 知的財産本部内

【氏名又は名称】 須澤 修

【選任した代理人】

【識別番号】 100107076

【住所又は居所】 長野県諏訪市大和3丁目3番5号 セイコーエプソン株式会社 知的財産本部内

【氏名又は名称】 藤網 英吉

特願 2 0 0 3 - 2 0 3 8 5 6

出 願 人 履 歴 情 報

識別番号

[0 0 0 0 0 2 3 6 9]

1. 変更年月日

1 9 9 0 年 8 月 2 0 日

[変更理由]

新規登録

住 所

東京都新宿区西新宿 2 丁目 4 番 1 号

氏 名

セイコーエプソン株式会社